PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-229040
(43)Date of publication of application : 24.08.2001

(51)Int.Cl. G06F 9/46
G06F 1/32
// G06F 15/78

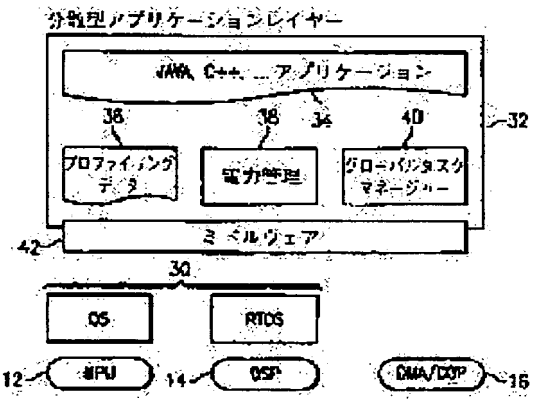
(21)Application number : 2000-366246
(22)Date of filing : 25.10.2000

(71)Applicant : TEXAS INSTR INC <TI>
(72)Inventor : CHAUVEL GERARD
DINVERNO DOMINIQUE BENOIT J

(30)Priority
Priority number : 1999 99402655 Priority date : 25.10.1999 Priority country : EP

(54) METHOD AND DEVICE FOR MANAGING INTELLIGENT POWER FOR DISTRIBUTED PROCESSING SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To manage a power in a circuit without having any significant effect on performance.
SOLUTION: This distributed processing system includes plural processing modules, for example, MPU 12, DSP 14, and co-processor/DMA channel 16. Then, a scenario to fulfill a prescribed power target, for example, to perform the maximum operation within the thermal constraint of a package or a target to use the minimum energy is prepared by using power management software 38 related with a profile 36 for various processing modules and tasks to be executed. The actual activity related with the tasks is monitored during the operation so that conformance with the target can be compensated. Also, the assignment of tasks can be dynamically changed so that conformance with the change of an environmental condition and the change of a task list can be obtained.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号
特開2001－229040
（P2001－229040A）
(43)公開日　平成13年８月24日(2001.8.24)

(51)Int.Cl.'		識別記号	FI		テーマコード'（参考）
Ｇ０６Ｆ	9/46	３６０	Ｇ０６Ｆ 9/46		３６０Ｂ
	1/32			15/78	５１０Ｐ
// Ｇ０６Ｆ	15/78	５１０		1/00	３３２Ｚ

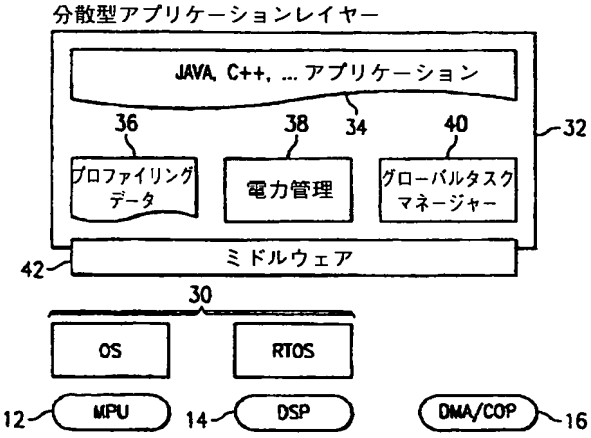
審査請求　未請求　請求項の数22　ＯＬ　外国語出願　（全 34 頁）

(21)出願番号　特願2000－366246(P2000－366246)
(22)出願日　平成12年10月25日(2000.10.25)
(31)優先権主張番号　９９４０２６５５．７
(32)優先日　平成11年10月25日(1999.10.25)
(33)優先権主張国　欧州特許庁（ＥＰ）

(71)出願人　590000879
テキサス　インスツルメンツ　インコーポレイテツド
アメリカ合衆国テキサス州ダラス，ノース　セントラルエクスプレスウエイ　13500
(72)発明者　ジェラール　ショベル
フランス国　アンチーブ、ピュ　ニュメロ　20、シュマン　デュ　バル　ボスクェ　292、バル　ボスクェ
(72)発明者　ドミニク　ブノワ　ジャック　ダンベルノ
フランス国　ピュヌーブ　－　ルーブ、シュマン　デバッス　ジヌスティエール　47
(74)代理人　100066692
弁理士　浅村　皓　（外３名）

(54)【発明の名称】　分散型処理システムのためのインテリジェント電力管理方法および装置

(57)【要約】　（修正有）
【課題】　性能に深刻な影響を与えることなく、回路内で電力を管理すること
【解決手段】　分散型処理システムは複数の処理モジュール、例えばＭＰＵ１２と、ＤＳＰ１４と、コプロセッサ／ＤＭＡチャンネル１６を含む。種々の処理モジュールおよび実行すべきタスクのためのプロフィル３６に関連する電力管理ソフトウェア３８を使って、所定の電力目的、例えばパッケージの熱的制約内で最大の作動を行ったり、または最小のエネルギーを使用する目的を満たすシナリオを作成する。目的との適合性を補償するよう、作動中にタスクに関連する実際のアクティビティをモニタする。環境条件の変化およびタスクリストの変化に適合するように、タスクの割り当てをダイナミックに変えることができる。



1

【特許請求の範囲】
【請求項１】　複数の処理モードを含むプロセッサにおけるタスクの実行を制御するための方法であって、
タスクに関連するアクティビティの確率値に基づき、消費情報を計算する工程と、
前記消費情報に応答し、前記複数の処理モジュールでタスクを実行する工程を含む、タスクの実行を制御するための方法。
【請求項２】　処理モジュールにおける実際のアクティビティの発生をモニタする工程と、
前記モニタ工程に基づき、タスクの実行を変更する工程とを更に含む、請求項１記載の方法。
【請求項３】　前記タスクの実行工程が、処理システムに関連する熱的制約内で最大の性能を発揮するよう、前記消費情報に応答して前記複数の処理モジュールでタスクを実行する工程を含む、請求項１または２記載の方法。
【請求項４】　前記タスクの実行工程が、可能な最小エネルギー消費量を使用してタスクを実行するように、前記消費情報に応答して前記複数の処理モジュールでタスクを実行する工程を含む、請求項１または２記載の方法。
【請求項５】　前記計算する工程が、
タスクの割り当てシナリオを発生する工程と、
タスク割り当てシナリオに対するアクティビティを推定する工程と、
前記アクティビティに関連した消費量を計算する工程を含む、請求項１または２記載の方法。
【請求項６】　タスク割り当てシナリオを発生する前記工程が、実行すべきタスクを記述するタスクリストおよびタスクを記述するタスクモデルを受信する工程を含む、請求項５記載の方法。
【請求項７】　タスクモデルが各タスクに対する初期推定値を含む、請求項６記載の方法。
【請求項８】　タスクモデルがタスクに関連する優先度の制約を更に含む、請求項７記載の方法。
【請求項９】　タスクモデルが前記タスクリスト内のタスクの１つ以上に関連する可能性のあるエネルギー低減に関する情報を含む、請求項８記載の方法。
【請求項１０】　前記計算する工程が、前記アクティビティに関連するエネルギー消費量を計算する工程を含む、請求項５記載の方法。
【請求項１１】　前記計算する工程が、前記アクティビティに関連する電力消費量を計算する工程を含む、請求項５記載の方法。
【請求項１２】　複数のタスクを実行するための１つ以上の処理モジュールを含み、前記処理サブシステムが、
タスクに関連するアクティビティの確率値に基づき、消費情報を計算し、
前記消費情報に応答し、前記処理モジュールでのタスクの実行を制御するために、電力管理機能を実行するようになっている処理装置。

2

【請求項１３】　アクティビティの発生を測定するためのカウンタを更に含み、前記電力管理機能が、
前記カウンタをモニタし、
前記カウンタ内の値に基づき、タスクの実行を変えるようになっている、請求項１２記載の処理装置。
【請求項１４】　前記電力管理機能が、処理システムに関連する熱的制約内で最大の性能を発揮するように、前記消費情報に応答し、処理モジュールでのタスクの実行を制御するようになっている、請求項１２または１３記載の処理装置。
【請求項１５】　前記電力管理機能が、可能な最小エネルギー消費量を使用してタスクを実行するように、前記消費情報に応答し、処理モジュールでのタスクの実行を制御するようになっている、請求項１２または１３記載の処理装置。
【請求項１６】　前記電力管理機能が、
タスクの割り当てシナリオを発生し、
タスク割り当てシナリオに対するアクティビティを推定し、
前記アクティビティに関連した消費量を計算することにより消費情報を計算するようになっている請求項１２または１３記載の処理装置。
【請求項１７】　前記電力管理機能が、実行すべきタスクを記述するタスクリストおよびタスクを記述するタスクモデルを受信することによりタスク割り当てシナリオを発生するようになっている、請求項１６記載の処理装置。
【請求項１８】　タスクモデルが各タスクに対する初期推定値を含む、請求項１７記載の処理装置。
【請求項１９】　前記タスクモデルがタスクに関連する優先度の制約を更に含む、請求項１８記載の処理装置。
【請求項２０】　前記タスクモデルが前記タスクリスト内のタスクの１つ以上に関連する可能なエネルギー低減に関する情報を含む、請求項１９記載の処理装置。
【請求項２１】　前記電力管理機能が、前記アクティビティに関連するエネルギー消費量を計算することにより消費量を計算する、請求項１６記載の処理装置。
【請求項２２】　前記電力管理機能が、前記アクティビティに関連する電力消費量を計算することにより消費量を計算する、請求項１６記載の処理装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は集積回路に関し、より詳細には、プロセッサにおける電力管理に関する。
【０００２】
【従来の技術】何年もの間、マイクロプロセッサユニット（ＭＰＵ）、コプロセッサおよびデジタル信号プロセッサ（ＤＳＰ）のための設計を含むプロセッサの設計

は、プロセッサの速度および機能を高めることに焦点が合わせられてきた。現在、電力消費量が深刻な問題となっている。速度および機能を深刻に損なうことなく、低電力消費量を維持することは、多くの設計において最も重要なこととなっている。多くのシステム、例えばスマートフォーン、携帯電話、ＰＤＡ（パーソナルデジタルアシスタント）およびハンドヘルドコンピュータは、比較的小型の電池によって作動されるので、多くのアプリケーションでは電力消費量が重要である。短期間の後でバッテリーを充電するのは不便であるので、これらシステムではバッテリーの寿命を最大にすることが望ましい。

【０００３】現在、電力消費量を最小にする方法として、スタティックな電力管理方法がある。すなわち、使用電力をより少なくする回路設計を行っている。あるケースでは、ダイナミックな電力管理を行っている。例えばクロック速度を遅くしたり、アイドル時間中に回路をディスエーブルしている。

【０００４】

【発明が解決しようとする課題】これら変更は重要であるが、特にデバイスの使用をより便利にする上で、サイズ、すなわちバッテリーサイズが重要であるシステムでは、電力管理の改善を続ける必要がある。

【０００５】電力全体を節約する外に、複雑な処理環境では集積回路から熱を散逸できる能力も１つの要素となっている。集積回路は所定の量の熱を散逸させるようになっており、高レベルの電流を引き出すためにタスクが集積回路上に多数のシステムを必要とする場合、回路が過熱し、システムが故障する可能性がある。

【０００６】将来、集積回路で実現されるアプリケーションはより複雑となり、単一の集積回路内に設けられたＭＰＵ、ＤＳＰ、コプロセッサおよびDMAチャンネルを含むマルチプロセッサ（以下、分散型処理システムと称す）により、マルチ処理を行う可能性がある。ＤＳＰは多数の同時アプリケーションをサポートする。アプリケーションの一部は特定のＤＳＰプラットフォームの専用ではないが、グローバルネットワーク、例えばインターネットからロードされる。従って、過熱を生じることなく、分散型処理システムが処理できるタスクは不確実となる。

【０００７】従って、性能に深刻な影響を与えることなく、回路内で電力を管理するための方法および装置に対するニーズが生じている。

【０００８】

【課題を解決するための手段】本発明は複数の処理モジュールを含むプロセッサ内でのタスクの実行を制御するための方法および装置を提供するものである。タスクに関連するアクティビティーに対する確率値に基づき、電力消費量情報が計算され、この電力消費量情報に応答する処理モジュールでタスクが実行される。

【０００９】本発明は従来技術よりも大きな利点を有する。第１に、完全にダイナミックな電力管理を可能にできることが挙げられる。処理システム内で実行されるタスクが変化するにつれ、電力管理ソフトウェアはスレッショルドを越えないように新しいシナリオを作成できる。更に、環境条件が変化するにつれ、例えばバッテリー電圧が低下するにつれ、電力管理ソフトウェアは条件を評価し直し、必要であればシナリオを変更できる。第２に、電力管理ソフトウェアが制御する種々のタスクに対して、この電力管理ソフトウェアがトランスペアレントであることが挙げられる。従って、特定のタスクが電力管理を可能にしない場合でも、この電力管理ソフトウェアは処理システムの電力能力と合致した態様でタスクを実行する責任を果たす。第３に、電力計算を行うのに使用されるプロフィルを変えることにより適合される、異なるハードウェアプラットフォーム、異なるハードウェアおよびタスクと共に電力管理ソフトウェアの作動全体を使用できる。

【００１０】以下、本発明およびその利点をより完全に理解できるよう、添付図面と関連して次の説明を参照する。

【００１１】

【発明の実施の形態】図面中の図１～９を参照すれば、本発明について最良に理解できよう。種々の図面のうちの同様な要素に対して同様な番号を使用する。

【００１２】図１は、ＭＰＵ１２、１つ以上のＤＳＰ１４および１つ以上のDMAチャンネルまたはコプロセッサ（DMA／コプロセッサ１６と一括して示す）を含む一般的な分散型処理システム１０の全体のブロック図を示す。この実施例では、ＭＰＵ１２はコア１８とキャッシュ２０とを含む。ＤＳＰ１４は処理コア２２とローカルメモリ２４とを含む（実際の実施例は別個の命令およびデータメモリを使用することもできるし、また統一された命令兼データメモリを使用することもできる）。メモリインターフェース２６は共用メモリ２８をＭＰＵ１２、ＤＳＰ１４またはDMA／コプロセッサ１６のうちの１つ以上に結合し、各プロセッサ（ＭＰＵ１２、ＤＳＰ１４）は実際の分散型処理システム内の自己のオペレーティングシステム（ＯＳ）またはリアルタイムのオペレーティングシステム（ＲＴＯＳ）下で完全に自律的に作動できるし、またＭＰＵ１２は共用リソースおよびメモリ環境を監督するグローバルＯＳを作動できる。

【００１３】図２は、分散型処理システム１０のためのソフトウェアのレイヤー図を示す。図１に示されるように、ＭＰＵ１２はＯＳを実行するが、他方、ＤＳＰ１４はＲＴＯＳを実行する。ＯＳおよびＲＴＯＳはソフトウェアのＯＳレイヤー３０を含む。分散型アプリケーションレイヤー３２はＪＡＶＡ（登録商標）、Ｃ＋＋および他のアプリケーション３４、プロファイリングデータ３６を使用する電力管理タスク３８およびグローバルタス

クスケジューラ４０を含む。ミドルウェアソフトウェアレイヤー４２はＯＳレイヤー３０と分散型アプリケーションレイヤー３２内のアプリケーションとの間の通信を行う。

【００１４】図１および２を参照し、分散型処理システム１０の作動について説明する。分散型処理システム１０は種々のタスクを実行できる。分散型処理システム１０のための代表的なアプリケーションとしては、スマートフォーンアプリケーション内に設けられることになろう。このスマートフォーンアプリケーションでは分散型処理システム１０が無線通信、ビデオおよびオーディオのデコンプレッションおよびユーザーインターフェース（例えばＬＣＤの更新、キーボードのデコード）を取り扱う。このアプリケーションでは、分散型処理システム１０内の異なる埋め込みシステムが異なる優先度の多数のタスクを実行する。一般には、ＯＳは種々の埋め込み型システムへの異なるタスクのタスクスケジュール設定を実行する。

【００１５】本発明は、タスクのスケジュール設定における基準としてエネルギー消費量を積分する。好ましい実施例では、タスクのリストを実行するための確率値に基づくシステムのシナリオを作成するために、電力管理アプリケーション３８および分散型アプリケーションレイヤー３２からのプロフィル３６を使用する。シナリオが所定の基準を満たさない場合、例えば電力消費量が過度に多い場合、新しいシナリオを発生する。許容できるシナリオを設定した後にＯＳレイヤーはシナリオで予想されたアクティビティが正確であったかどうかを検証するように、ハードウェアのアクティビティをモニタする。

【００１６】許容できるタスクスケジュール設定シナリオのための基準はデバイスの性質に応じて変わり得る。移動デバイスのための重要な基準は最小エネルギー消費量である。上記のように電子通信デバイスが更に小型化されるにつれ、より小型のバッテリーの割り当てによってエネルギー消費量にプレミアムが課される。デバイス作動中の多くのケースでは、特にバッテリーが低レベルに達する際に、電力を低減するよう、タスクのためのエネルギー低減作動モードを許可できる。例えば画質を代償に、ＬＣＤのリフレッシュレートを下げて電力を低減する。別のオプションとしては、性能が低速化することを代償に、分散型処理システム１０のＭＩＰ（１秒あたりの１００万回数の命令の実行回数を単位とする）を低減し、電力を低減することが挙げられる。電力管理ソフトウェア３８はデバイスの許容可能な作動に達するよう、低下した性能の異なる組み合わせを使用する別のシナリオを解析できる。

【００１７】電力管理における別の目的は、所定の電力限界設定量に対する最大のＭＩＰまたは最低のエネルギーを探すことである。

【００１８】図３ａおよび３ｂは分散型処理システム１０が平均的な電力差散逸限界を越えないように、電力管理アプリケーション３８を使用する一例を示す。図３ａではＤＳＰ１４、ＤＭＡ１６およびＭＰＵ１２が多数のタスクを同時に実行中である。時間ｔ１において、３つの埋め込み型システムの平均電力散逸量は分散型処理システム１０に課された平均的限界を越えている。図３ｂは同じタスクを実行する場合のシナリオを示している。しかしながら、許容可能な電力散逸プロフィルを維持するために、ＤＭＡおよびＤＳＰのタスクを完了するまで、ＭＰＵのタスクを遅延している。

【００１９】図４ａは、電力管理タスク３８の第１実施例の作動を示すフローチャートである。ブロック５０において、グローバルスケジューラ４０により電力管理タスクが呼び出され、このタスクはＭＰＵ１２またはＤＳＰ１４の１つによって実行できる。スケジューラは入力されたアプリケーションを評価し、これを関連する優先度および排除ルールに関連する複数のタスクに分割する。タスクリスト５２は、例えばオーディオ／ビデオデコーディング、ディスプレイ制御、キーボード制御、キャラクター認識などを含むことができる。ステップ５４において、タスクモデルファイル５６および受け入れられたエネルギー低減ファイル５８を考慮して、タスクリスト５２を評価する。このタスクモデルファイル５６は分散型アプリケーションレイヤー３２のプロフィル３６の一部である。タスクモデルファイル５６はタスクリスト内の各リストに異なるモデルを割り当てる、先に発生されたファイルである。各モデルはデータの集合であり、このデータの集合は実験的に、またはコンピュータ補助ソフトウェアデザイン技術によって誘導でき、関連するタスクの特性、例えばレイテンシーの制約、優先度、データフロー、基準プロセッサ速度における初期エネルギー予想値、エネルギー低減の影響およびＭＩＰおよび時間に応じた所定のプロセッサにおける実行プロフィルを定める。エネルギー低減リスト５８はシナリオを発生する際に使用できる種々のエネルギー低減方法を定める。

【００２０】タスクリストを変更するごとに（例えば新しいタスクを作成したり、タスクを削除するごとに）、またはリアルタイムイベントが発生する時に、ステップ５４におけるタスクリスト５２およびタスクモデル５６に基づき、シナリオを作成する。このシナリオはモジュールに種々のタスクを割り当て、タスクを実行する優先度を設定する優先度情報を与える。タスクのエネルギー推定値から基準速度におけるシナリオのエネルギー推定値５９を計算できる。必要であるか、または望ましい場合に、タスクを低減してもよい。すなわち、タスクのフルバージョンに対し、ほとんどリソースを使用しないタスクのモードに置換できる。このシナリオから、ブロック６０でアクティビティ推定値を発生する。このアクテ

ィビティ推定値は（分散型アクティビティレイヤー３２のプロファイリングデータ３６からの）タスクアクティビティプロフィル６２および（分散型アプリケーションレイヤー３２のプロファイリングデータ３６からの）ハードウェアアーキテクチャモデル６４を使用し、シナリオから生じるハードウェアアクティビティに対する確率値を発生する。この確率値は各モジュールの待機／実行時間の比率（有効ＭＨｚ）、キャッシュおよびメモリへのアクセス、Ｉ／Ｏ切り替えレートおよびＤＭＡフローリクエストおよびデータボリュームを含む。熱時間定数に一致する期間Ｔを使用すると、基準プロセッサ速度およびステップ６０で誘導された平均アクティビティ（特にプロセッサの有効速度）から熱パッケージモデルに匹敵する平均電力散逸量を計算することができる。電力値がパッケージ熱モデル７２に記載されたスレッショルドを越える場合、判断ブロック７４でシナリオを拒否する。この場合、ブロック５４で新しいシナリオを作成し、ステップ６０、６６および７０を繰り返す。スレッショルドを越えない場合、シナリオを使ってタスクリストを実行する。

【００２１】シナリオが定めるタスクの作動中に、ＯＳおよびＲＴＯＳはハードウェア内に組み込まれたカウンタ７８を使ってブロック７６でそれぞれのモジュールによるアクティビティを追跡する。分散型処理システム１０のモジュールにおける実際のアクティビティはブロック６０で推定されるアクティビティと異なることがある。ハードウェアカウンタからのデータは測定されたアクティビティ値を発生するようにＴ時間方法に基づき、モニタされる。これら測定されたアクティビティ値は、この時間の間のエネルギー値、従って上記のようにブロック６６における平均電力値を計算するのにブロック６６で使用され、ブロック７２においてパッケージ熱モデルと比較される。測定された値がスレッショルド値を越える場合、ブロック５４で新しいシナリオが作成される。測定されたアクティビティ値を連続的にモニタすることにより、所定の限度内に留まるように、または変化する環境条件に合わせるように、シナリオをダイナミックに変更できる。

【００２２】チップに対する時間Ｔにわたる総エネルギー消費量は次のように計算される。

【００２３】

【数１】

$$E = \int_T \sum_{\text{modules}} \left[\alpha \cdot Cpd \cdot f \cdot V_{dd}^2\right] dt = \sum_{\text{modules}} \left[\sum_T (\alpha \cdot)\right] Cpd \cdot f \cdot V_{dd}^2$$

【００２４】ここで、ｆは周波数であり、V_{dd}は供給電圧であり、αは確率（またはこの図のブロック７６を参照した説明を参照）アクティビティである。換言すれば、下記の式は等価消費キャパシタンスＣｐｄを特徴とする特定のハードウェアモジュールに対応するエネルギーである。

【００２５】

【数２】

$$\sum_T (\alpha) * Cpd * f * V_{dd}^2$$

【００２６】ここで、カウンタ値は次の式である。

【００２７】

【数３】

$$\sum_T (\alpha)$$

【００２８】ＥはＴ時間内で散逸される、分散型処理システム１０におけるすべてのモジュールに対するエネルギーの総和である。平均システム電力散逸量はＷ＝Ｅ／Ｔである。好ましい実施例では、測定エネルギー消費量および確率エネルギー消費量を計算し、時間Ｔにわたるエネルギー消費量から平均電力散逸量を誘導する。ほとんどのケースにおいて、エネルギー消費情報をより容易に入手できる。しかしながら、測定された電力消費量および確率電力消費量から電力散逸量を計算することも可能である。

【００２９】図４ｂは、電力管理タスク３８の第２実施例の作動を示すフローチャートである。図４ｂのフローは、図４１のフローと同じであるが、次の点が異なる。すなわち新しいシナリオを選択する代わりに、ステップ５０においてシナリオ作成アルゴリズムを呼び出す際に（新しいタスク、タスクの削除、リアルタイム事象の時に）、性能の制約に合致するｎ個の異なるシナリオをステップ５４および５９で予め計算し、記憶し、ダイナミックループ内の演算回数を低減し、トラッキングループ内で計算された電力によりブロック７４内で現在のシナリオを拒否することになる場合、より高速の適応化を実行する点が異なっている。図４ｂでは、シナリオを拒否する場合、ブロック６５で別の予め計算されたシナリオを選択する。拒否しない場合、動作は図４ａに示されたものと同一である。

【００３０】図５～８は、図３の種々のブロックの作動をより詳細に示す図である。図５には、シナリオ作成システムブロック５４が示されている。このブロックでは、シナリオを発生するのにタスクリスト５２、タスクモデル５６および可能性のあるタスク低減５８のリストを使用する。タスクリストは分散型処理システム１０でどのタスクを実行すべきかに依存している。図５の例では、３つのタスク、すなわちＭＰＥＧ４デコード、無線モデムデータ受信およびキーボードイベントモニタのタスクが示されている。実際の実現例では、これらタスクは任意の数のソースから得られる。タスクモデルはシナリオを作成する際に考慮すべき条件、例えばレイテンシーおよび優先度の制約、データフロー、初期エネルギー推定値およびエネルギー低減の影響を定めている。この

ブロックでは他の条件も使用できる。シナリオ作成システムブロックの出力はシナリオ８０であり、このシナリオは種々のタスクとモジュールとを関連づけ、タスクの各々に優先度を割り当てる。図５に示された例では、例えばＭＰＥＧ４は１６の優先度を有し、無線モデムタスクは４の優先度を有する。

【００３１】ブロック５４で作成されるシナリオは多数の異なる検討事項に基づいてもよい。例えばシナリオはパッケージの熱的制約内で最大の性能を発揮することに基づいて作成できる。これとは別に、シナリオは可能な最小のエネルギーを使用することに基づくことができる。最適なシナリオはデバイスの作動中に変わることができる。例えばバッテリーが完全に充電された状態では、デバイスは最大性能レベルで作動できる。バッテリー内の電力が設定されたレベル以下に減少するにつれ、デバイスは作動を維持する可能な最小電力レベルで作動できる。

【００３２】ブロック５４からのシナリオ８０は、図６に示されるアクティビティ推定ブロック６０によって使用される。このブロックは分散型処理システム１０における電力利用率に影響する種々のパラメータに対する確率計算を実行する。タスクアクティビティプロフィル６２およびハードウェアアーキテクチャモデル６４に関連して、アクティビティ確率推定値が発生される。タスクアクティビティプロフィルはデータアクセスのタイプ（ロード／記憶）および異なるメモリに対する発生、コードプロフィル、例えばタスクで使用されるブランチおよびループ、並びにタスク内の命令に対する命令ごとのサイクルに関するデータを含む。ハードウェアアーキテクチャモデル６４は、システムのレイテンシーに対するタスクアクティビティプロフィル６２の影響をある方法で記述しており、これにより、推定されるハードウェアアクティビティ（例えばプロセッサの実行／待機時間の比率）の計算が可能となっている。このモデルはタスクを実行するハードウェアの特性、例えばキャッシュのサイズ、種々のバスの幅、Ｉ／Ｏピンの数、キャッシュがライトスルーであるか、またはライトバックであるか、使用されるメモリのタイプ（ダイナミック、スタティック、フラッシュなど）およびモジュールで使用されるクロック速度を考慮している。一般に、このモデルは異なるパラメータ、キャッシャブル、非キャッシャブルデータ、読み出し／書き込みアクセスシェア、命令ごとのサイクル数などに対するＭＰＵおよびＤＳＰの有効周波数変化を示す曲線群から構成できる。図６の図示された実施例では、各モジュールの有効周波数に対する値、メモリアクセスの回数、Ｉ／Ｏ切り替えレートおよびＤＭＡフローを計算する。電力に影響する他の要因も計算できる。

【００３３】図８には、電力計算ブロック６６が示されている。このブロックでは種々のエネルギー値、従って

期間Ｔにわたる電力値を計算するのに、ブロック６０からの確率アクティビティまたはブロック７６からの測定アクティビティを使用する。電力値は分散型処理システム１０のハードウェアデザインに固有のハードウェア電力プロフィルに関連して計算される。このハードウェアプロフィルは、各モジュールのためのＣｐｄ、論理デザインスタイル（Ｄタイプのフリップフロップ、ラッチ、ゲート制御されるクロックなど）、電源電圧および出力端にかかる容量性負荷を含むことができる。集積モジュールに対し、更に外部メモリまたは他の外部デバイスに対しても電力計算をすることができる。

【００３４】図８にはアクティビティ測定およびモニタブロック７６が示されている。種々のモジュールでのアクティビティ、例えばキャッシュミス、ＴＬＢ（変換ルックアサイドバッファ）ミス、非キャッシャブルメモリアクセス、待機時間、異なるリソースのための読み出し／書き込みリクエスト、メモリオーバーヘッドおよび温度を測定するよう、分散型処理システム１０全体にわたってカウンタが構成さている。アクティビティ測定およびモニタブロック７６は、各モジュールの有効周波数、メモリアクセスの数、Ｉ／Ｏ切り替えレートおよびＤＭＡフローの値を出力する。特定の実現例では、他の値を測定してもよい。このブロックの出力は電力計算ブロック６６へ送られる。

【００３５】図９は、電力／エネルギー管理ソフトウェアを使用する分散型処理システム１０の一例を示す。この例では、分散型処理システム１０はＯＳを実行するＭＰＵ１２と、２つのＤＳＰ１４（別々にＤＳＰ１　１４ａおよびＤＳＰ２　１４ｂと表示される）を含み、各ＤＳＰはそれぞれのＲＴＯＳを実行するようになっている。各モジュールはモニタタスク８２を実行し、これにより分散型処理システム１０全体にわたる種々のアクティビティカウンタ７８内の値をモニタするようになっている。ＤＳＰ１４ａでは電力計算タスクが実行され、種々のモニタタスクは関連するアクティビティカウンタ７８からデータを検索し、情報をＤＳＰ１４ａへ送り、測定されたアクティビティに基づき電力値を計算する。電力管理タスク、例えば電力計算タスク８４およびモニタタスク８２は他のアプリケーションタスクと共に実行できる。

【００３６】好ましい実施例では、電力管理タスク３８およびプロフィル３６はＪＡＶＡリアルタイム環境内でＪＡＶＡクラスパッケージとして実現される。

【００３７】本発明は、従来技術よりも大きな利点を有する。第１に完全にダイナミックな電力管理が可能である。分散型処理システム１０内で実行されるタスクが変化するにつれ、電力管理はスレッショルド値を越えないように新しいシナリオを作成できる。更に、環境条件が変化するにつれ、例えばバッテリー電圧が低下するにつれ、電力管理ソフトウェアが条件を評価し直し、必要な

場合にシナリオを変えることができる。例えばＶｄｄを公称値に維持できない点までバッテリー電圧（電源電圧）が低下した場合、より低いＶｄｄで分散型処理システム１０を作動できる、より低い周波数を設定でき、より低い周波数を考慮した新しいシナリオを作成できる。ある状況では、より低い周波数を補償するように、より多くのエネルギー低減法を導入できる。しかしながら、通常不充分となる電源電圧にもかかわらず、周波数を低くすれば、デバイスの作動を持続することができる。更に、より低い周波数が認められる場合、比較的低いアクティビティの期間中に電力を節約するように（スイッチングモードの電源を利用できる場合に）、より低いＶｄｄでデバイスを作動できる。

【００３８】電力管理ソフトウェアは、このソフトウェアが制御する種々のタスクに対してトランスペアレントとなっている。従って、特定のタスクが電力管理を考慮していない場合でも、電力管理ソフトウェアは分散型処理システム１０の電力容量と合致する態様でタスクを実行する責任を果たす。

【００３９】異なるハードウェアプラットフォーム、プロフィル３６を変えることによって適合される異なるハードウェアおよびタスクと共に、電力管理ソフトウェアの作動全体を利用できる。

【００４０】以上の発明の詳細な説明は、所定の実施例を説明するものであったが、当業者にはこれら実施例の種々の変形例だけでなく、別の実施例を考え付くことができよう。本発明は特許請求の範囲に入る変形例または別の実施例を含むのものである。

【図面の簡単な説明】

【図１】分散型処理システムのブロック図を示す。

【図２】分散型処理システムのソフトウェアレイヤー図を示す。

【図３】分散型処理システムのための電力管理の利点を示す一例を示す。

【図４ａ】図２の電力管理ソフトウェアの作動に関する好ましい第１実施例の作動を示すフローチャートである。

【図４ｂ】図２の電力管理ソフトウェアの作動に関する好ましい第２実施例の作動を示すフローチャートである。

【図５】図４のシナリオ作成システムのブロックを示す。

【図６】図４のアクティビティ推定ブロックを示す。

【図７】図４の電力計算ブロックを示す。

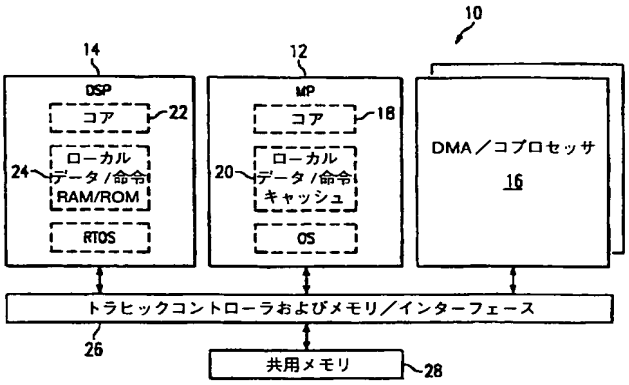
【図８】図４のアクティビティ測定およびモニタブロックを示す。

【図９】アクティビティカウンタを備えた分散型処理システムを示すブロック図である。

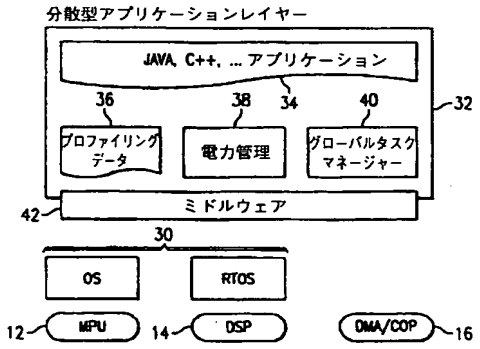
【符号の説明】

１０　分散型処理システム
１２　ＭＰＵ
１４　ＤＳＰ
１６　コプロセッサ／ＤＭＡチャンネル
３６　プロフィル
３８　電力管理ソフトウェア

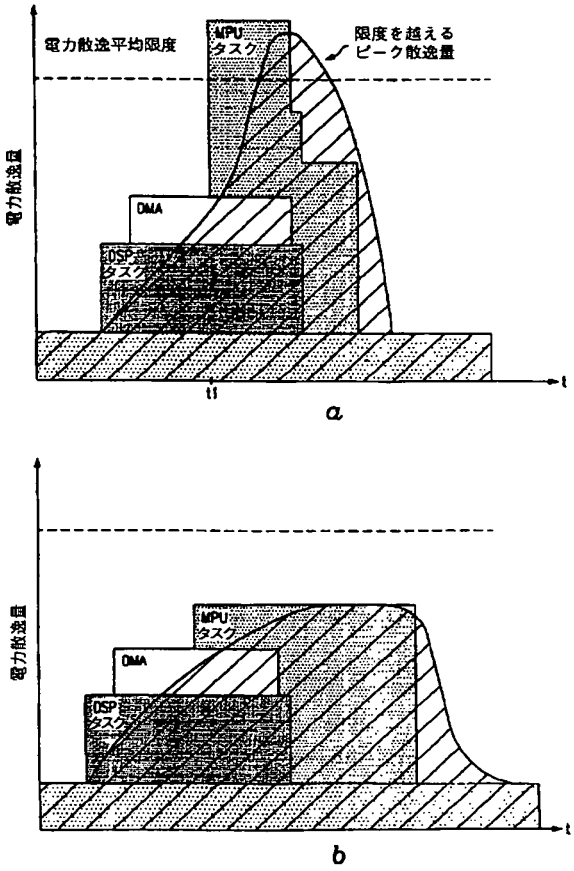
【図１】



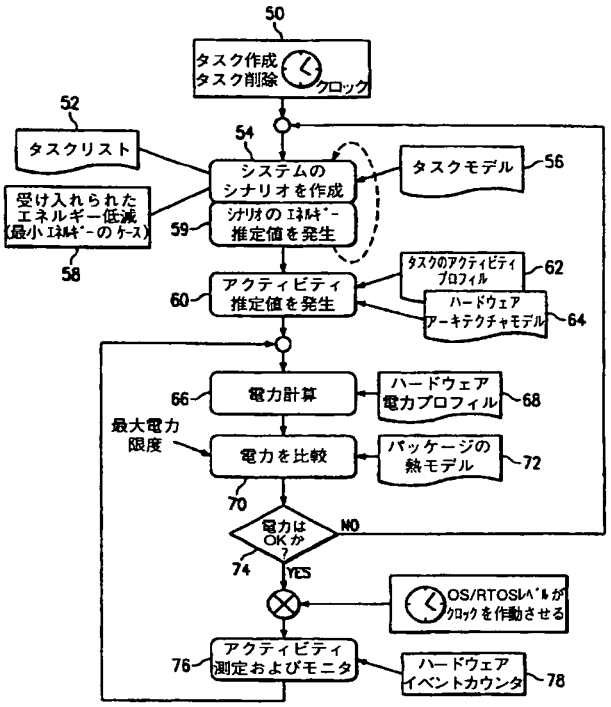
【図２】



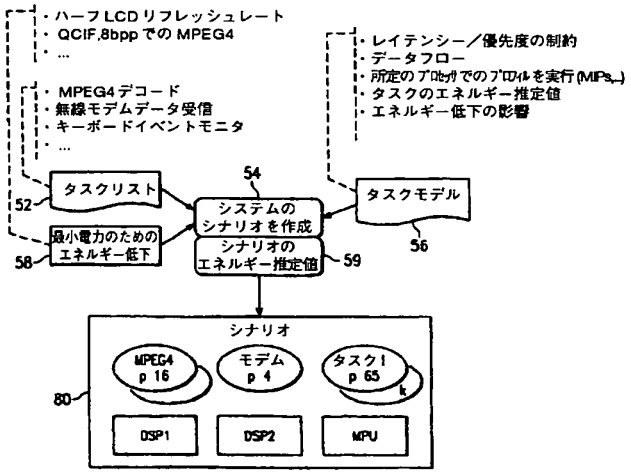
【図３】


電力散逸平均限度
MPU
タスク
限度を越える
ピーク散逸量
電力散逸量
DMA
DSP
タスク
t1
t
a
MPU
タスク
DMA
DSP
タスク
t
b

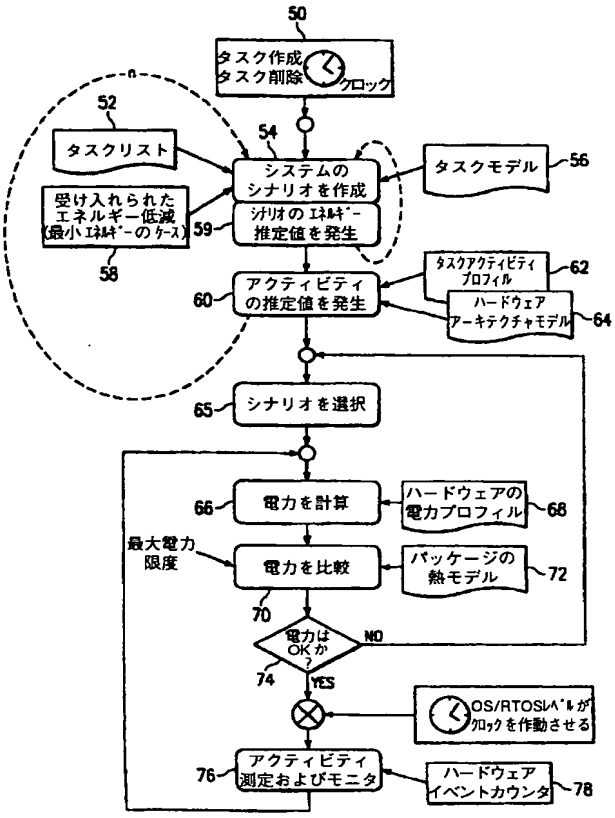
【図４ａ】


50
タスク作成
タスク削除
クロック
52
タスクリスト
54
システムの
シナリオを作成
タスクモデル
56
受け入れられた
エネルギー低減
(最小エネルギーのケース)
58
シナリオのエネルギー
推定値を発生
59
タスクのアクティビティ
プロフィル
62
アクティビティ
推定値を発生
60
ハードウェア
アーキテクチャモデル
64
電力計算
66
ハードウェア
電力プロフィル
68
最大電力
限度
電力を比較
パッケージの
熱モデル
72
70
電力は
OKか
?
NO
74
YES
OS/RTOSレベルが
クロックを作動させる
アクティビティ
測定およびモニタ
76
ハードウェア
イベントカウンタ
78

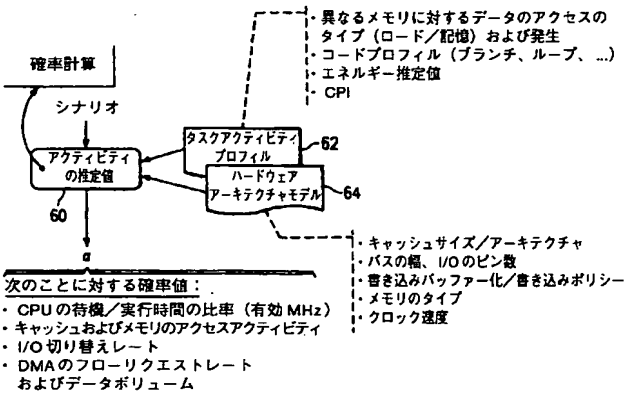
【図５】


・ハーフLCDリフレッシュレート
・QCIF,8bppでのMPEG4
・...
・MPEG4デコード
・無線モデムデータ受信
・キーボードイベントモニタ
・...
・レイテンシー／優先度の制約
・データフロー
・所定のプロセッサでのプロフィルを実行(MIPs,..)
・タスクのエネルギー推定値
・エネルギー低下の影響
52
タスクリスト
54
システムの
シナリオを作成
タスクモデル
56
58
最小電力のための
エネルギー低下
シナリオの
エネルギー推定値
59
シナリオ
80
MPEG4
p 16
モデム
p 4
タスク1
p 65
DSP1
DSP2
MPU

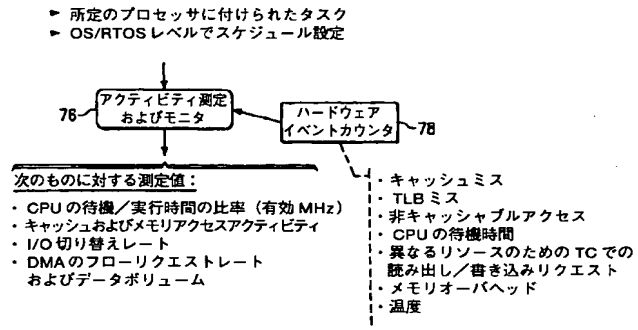
【図４ｂ】


50
タスク作成
タスク削除
クロック
52
タスクリスト
54
システムの
シナリオを作成
タスクモデル
56
受け入れられた
エネルギー低減
(最小ｴﾈﾙｷﾞｰのｹｰｽ)
59
ｼﾅﾘｵのｴﾈﾙｷﾞｰ
推定値を発生
58
タスクアクティビティ
プロフィル
62
アクティビティ
の推定値を発生
60
ハードウェア
アーキテクチャモデル
64
65
シナリオを選択
66
電力を計算
ハードウェアの
電力プロフィル
68
最大電力
限度
電力を比較
パッケージの
熱モデル
72
70
電力は
OKか
?
NO
74
YES
OS/RTOSﾚﾍﾞﾙが
ｸﾛｯｸを作動させる
76
アクティビティ
測定およびモニタ
ハードウェア
イベントカウンタ
78

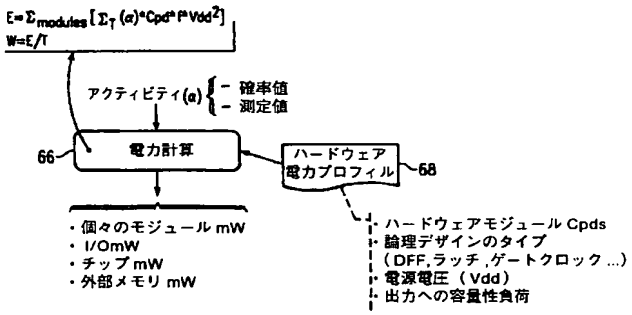
【図６】


確率計算
シナリオ
アクティビティ
の推定値
60
タスクアクティビティ
プロフィル
62
ハードウェア
アーキテクチャモデル
64
次のことに対する確率値：
・CPUの待機／実行時間の比率（有効 MHz）
・キャッシュおよびメモリのアクセスアクティビティ
・I/O切り替えレート
・DMAのフローリクエストレート
およびデータボリューム
・異なるメモリに対するデータのアクセスの
タイプ（ロード／記憶）および発生
・コードプロフィル（ブランチ、ループ、...）
・エネルギー推定値
・CPI
・キャッシュサイズ／アーキテクチャ
・バスの幅、I/Oのピン数
・書き込みバッファー化／書き込みポリシー
・メモリのタイプ
・クロック速度

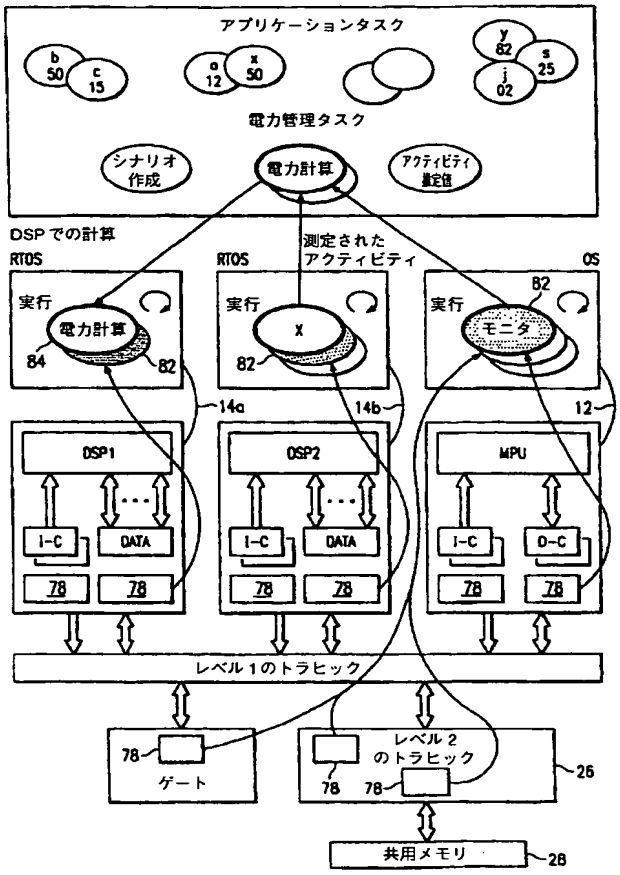
【図８】


► 所定のプロセッサに付けられたタスク
► OS/RTOSレベルでスケジュール設定
76
アクティビティ測定
およびモニタ
ハードウェア
イベントカウンタ
78
次のものに対する測定値：
・CPUの待機／実行時間の比率（有効 MHz）
・キャッシュおよびメモリアクセスアクティビティ
・I/O切り替えレート
・DMAのフローリクエストレート
およびデータボリューム
・キャッシュミス
・TLBミス
・非キャッシャブルアクセス
・CPUの待機時間
・異なるリソースのためのTCでの
読み出し／書き込みリクエスト
・メモリオーバヘッド
・温度

【図７】


E=Σmodules[Στ(α)*Cpd*f*Vdd2]
W=E/T
アクティビティ(α)
確率値
測定値
66
電力計算
ハードウェア
電力プロフィル
68
・個々のモジュール mW
・I/OmW
・チップ mW
・外部メモリ mW
・ハードウェアモジュール Cpds
・論理デザインのタイプ
（DFF,ラッチ,ゲートクロック...）
・電源電圧（Vdd）
・出力への容量性負荷

【図９】


アプリケーションタスク
b 50
c 15
a 12
x 50
y 82
s 25
j 02
電力管理タスク
シナリオ作成
電力計算
アクティビティ測定値
DSPでの計算
測定されたアクティビティ
RTOS
RTOS
OS
実行
電力計算
84
82
実行
X
82
実行
82
モニタ
14a
14b
12
DSP1
DSP2
MPU
I-C
DATA
I-C
DATA
I-C
D-C
78
78
78
78
78
78
レベル１のトラヒック
78
ゲート
レベル２のトラヒック
78
78
26
共用メモリ
28

【外国語明細書】

INTELLIGENT POWER MANAGEMENT FOR DISTRIBUTED PROCESSING SYSTEMS

BACKGROUND OF THE INVENTION

1. TECHNICAL FIELD

This invention relates in general to integrated circuits and, more particularly, to managing power in a processor.

2. DESCRIPTION OF THE RELATED ART

For many years, the focus of processor design, including designs for microprocessor units (MPUs), co-processors and digital signal processors (DSPs), has been to increase the speed and functionality of the processor. Presently, power consumption has become a serious issue. Importantly, maintaining low power consumption, without seriously impairing speed and functionality, has moved to the forefront in many designs. Power consumption has become important in many applications because many systems, such as smart phones, cellular phones, PDAs (personal digital assistants), and handheld computers operate from a relatively small battery. It is desirable to maximize the battery life in these systems, since it is inconvenient to recharge the batteries after short intervals.

Currently, approaches to minimizing power consumption involve static power management; i.e., designing circuits which use less power. In some cases, dynamic actions have been taken, such as reducing clock speeds or disabling circuitry during idle periods.

While these changes have been important, it is necessary to continuously improve power management, especially in systems where size and, hence, battery size, is important to the convenience of using a device.

In addition to overall power savings, in a complex processing environment, the ability to dissipate heat from the integrated circuit becomes a factor. An integrated circuit will be designed to dissipate a certain amount of heat. If tasks require multiple systems on the integrated circuit to draw high levels of current, it is possible that the circuit will overheat, causing system failure.

In the future, applications executed by integrated circuits will be more complex and will likely involve multiprocessing by multiple processors, including MPUs, DSPs, coprocessors and DMA channels in a single integrated circuit (hereinafter, a "distributed processing system"). DSPs will evolve to support multiple, concurrent applications, some of which will not be dedicated to a specific DSP platform, but will be loaded from a global network such as the Internet. Accordingly, the tasks that a distributed processing system will be able to handle without overheating will become uncertain.

Accordingly, a need has arisen for a method and apparatus for managing power in a circuit without seriously impacting performance.

BRIEF SUMMARY OF THE INVENTION

The present invention provides a method and apparatus for controlling the execution of tasks in a processor comprising a plurality of processing modules. Consumption information is calculated based on probabilistic values for activities associated with the tasks. Tasks are then executed on the processing modules responsive to the consumption information.

The present invention provides significant advantages over the prior art. First, it provides for a fully dynamic power management. As the tasks executed in the processing system change, the power management software can build new scenarios to ensure that thresholds are not exceeded. Further, as environmental conditions change, such as battery voltages dropping, the power management software can re-evaluate conditions and change scenarios, if necessary. Second, the power management software is transparent to the various tasks that it controls. Thus, even if a particular task does not provide for any power management, the power management software assumes responsibility for executing the task in a manner that is consistent with the power capabilities of the processing system. Third, the overall operation of the power management software can be used with different hardware platforms, with different hardware and tasks accommodated by changing profiles used for making the power calculations.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

For a more complete understanding of the present invention, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, in which:

Figure 1 illustrates a block diagram of a distributed processing system;

Figure 2 illustrates a software layer diagram for the distributed processing system;

Figure 3 illustrates an example showing the advantages of power management for a distributed processing system;

Figures 4a and 4b illustrate flow diagrams showing preferred embodiments for the operation of the power management software of Figure 2;

Figure 5 illustrates the building system scenario block of Figure 4;

Figure 6 illustrates the activities estimate block of Figure 4;

Figure 7 illustrates the power compute block of Figure 4;

Figure 8 illustrates the activity measure and monitor block of Figure 4;

Figure 9 illustrates a block diagram showing the distributed processing system with activity counters.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is best understood in relation to Figures 1-9 of the drawings, like numerals being used for like elements of the various drawings.

Figure 1 illustrates a general block diagram of a general distributed processing system 10, including an MPU 12, one or more DSPs 14 and one or more DMA channels or coprocessors (shown collectively as DMA/Coprocessor 16). In this embodiment, MPU 12 includes a core 18 and a cache 20. The DSP 14 includes a processing core 22 and a local memory 24 (an actual embodiment could use separate instruction and data memories, or could use a unified instruction and data memory). A memory interface 26 couples a shared memory 28 to one or more of the MPU 12, DSP 14 or DMA/Coprocessor 16. Each processor (MPU 12, DSPs 14) can operate in full autonomy under its own operating system (OS) or real-time operating system (RTOS) in a real distributed processing system, or the MPU 12 can operate the global OS that supervises shared resources and memory environment.

Figure 2 illustrates a software layer diagram for the distributed processing system 10. As shown in Figure 1, the MPU 12 executes the OS, while the DSP 14 executes an RTOS. The OS and RTOSs comprise the OS layer 30 of the software. A distributed application layer 32 includes JAVA, C++ and other applications 34, power management tasks 38 which use profiling data 36 and a global tasks scheduler 40. A middleware software layer 42 communicates between the OS layer 30 and the applications in the distributed application layer 32.

Referring to Figures 1 and 2, the operation of the distributed processing system 10 is discussed. The distributed processing system 10 can execute a variety of tasks. A typical application for the distributed processing system 10 would be in a smartphone application where the distributed processing system 10 handles wireless communication, video and audio decompression, and user interface (i.e., LCD update, keyboard decode). In this application, the different embedded systems in the distributed processing system 10 would be executing multiple tasks of different priorities. Typically, the OS would perform the task scheduling of different tasks to the various embedded systems.

The present invention integrates energy consumption as a criterion in scheduling tasks. In the preferred embodiment, the power management application 38 and profiles 36 from the

distributed applications layer 32 are used to build a system scenario, based on probabilistic values, for executing a list of tasks. If the scenario does not meet predetermined criteria, for example if the power consumption is too high, a new scenario is generated. After an acceptable scenario is established, the OS layer monitors the hardware activity to verify that the activity predicted in the scenario was accurate.

The criteria for an acceptable task scheduling scenario could vary depending upon the nature of the device. One important criterion for mobile devices is minimum energy consumption. As stated above, as electronic communication devices are further miniaturized, the smaller battery allocation places a premium on energy consumption. In many cases during the operation of a device, a degraded operating mode for a task may be acceptable in order to reduce power, particularly as the batteries reach low levels. For example, reducing the LCD refresh rate will decrease power, albeit at the expense of picture quality. Another option is to reduce the MIPs (millions of instructions per second) of the distributed processing system 10 to reduce power, but at the cost of slower performance. The power management software 38 can analyze different scenarios using different combinations of degraded performance to reach acceptable operation of the device.

Another objective in managing power may be to find the highest MIPs, or lowest energy for a given power limit setup.

Figures 3a and 3b illustrate an example of using the power management application 38 to prevent the distributed processing system 10 from exceeding an average power dissipation limit. In Figure 3a, the DSP 14, DMA 16 and MPU 12 are concurrently running a number of tasks. At time t1, the average power dissipation of the three embedded systems exceeds the average limit imposed on the distributed processing system 10. Figure 3b illustrates a scenario where the same tasks are executed; however, an MPU task is delayed until after the DMA and DSP tasks are completed in order to maintain an acceptable average power dissipation profile.

Figure 4a illustrates a flow chart describing operation of a first embodiment of the power management tasks 38. In block 50, the power management tasks are invoked by the global scheduler 40, which could be executed on the MPU 12 or one of the DSPs 14; the scheduler evaluate the upcoming application and splits it into tasks with associated precedence and

exclusion rules. The task list 52 could include, for example, audio/video decoding, display control, keyboard control, character recognition, and so on. In step 54, the task list 52 is evaluated in view of the task model file 56 and the accepted degradations file 58. The task model file 56 is part of the profiles 36 of the distributed applications layer 32. The task model file 56 is a previously generated file that assigns different models to each task in the task list. Each model is a collection of data, which could be derived experimentally or by computer aided software design techniques, which defines characteristics of the associated task, such as latency constraints, priority, data flows, initial energy estimate at a reference processor speed, impacts of degradations, and an execution profile on a given processor as a function of MIPs and time. The degradation list 58 sets forth the variety of degradations that can be used in generating the scenario.

Each time the task list is modified (i.e., a new task is created or a task is deleted) or when a real time event occur, based on the task list 52 and the task model 56 in step 54, a scenario is built. The scenario allocates the various tasks to the modules and provides priority information setting the priority with which tasks are executed. A scenario energy estimate 59 at a reference speed can be computed from the tasks' energy estimate. If necessary or desirable, tasks may be degraded; i.e., a mode of the task that uses fewer resources may be substituted for the full version of a task. From this scenario, an activities estimate is generated in block 60. The activities estimate uses task activity profiles 62 (from the profiling data 36 of the distributed application layer 32) and a hardware architectural model 64 (also from the profiling data 36 of the distributed application layer 32) to generate probabilistic values for hardware activities that will result from the scenario. The probabilistic values include each module's wait/run time share (effective MHz), accesses to caches and memories, I/O toggling rates and DMA flow requests and data volume. Using a period T that matches the thermal time constant, from the energy estimate 59 at a reference processor speed and the average activities derived in step 60 (particularly, effective processors speeds), it is possible to compute an average power dissipation that will be compared to thermal package model. If the power value exceeds any thresholds set forth in the package thermal model 72, the scenario is rejected in decision block 74. In this case, a new scenario is built in block 54 and steps 60, 66 and 70 are repeated. Otherwise, the scenario is used to execute the task list.

During operation of the tasks as defined by the scenario, the OS and RTOSs track activities by their respective modules in block 76 using counters 78 incorporated in the hardware. The actual activity in the modules of the distributed processing system 10 may vary from the activities estimated in block 60. The data from the hardware counters are monitored on a T periodic basis to produce measured activity values. These measured activity values are used in block 66 to compute an energy value for this period, and hence, an average power value in block 66, as described above, and are compared to the package thermal model in block 72. If the measured values exceed thresholds, then a new scenario is built in block 54. By continuously monitoring the measured activity values, the scenarios can be modified dynamically to stay within predefined limits or to adjust to changing environmental conditions.

Total energy consumption over T for the chip is calculated as:

$$E = \int_T \sum_{modules} \left[\alpha \cdot Cpd \cdot f \cdot V_{dd}^2\right] dt = \sum_{modules} \left[\sum_T (\alpha \cdot)\right] \cdot Cpd \cdot f \cdot V_{dd}^2$$

where, f is the frequency, V_{dd} is the supply voltage and α is the probabilistic (or measured, see discussion in connection with block 76 of this figure) activity. In other words, $\sum_T (\alpha) * Cpd * f * V_{dd}^2$ is the energy corresponding to a particular hardware module characterized by equivalent dissipation capacitance Cpd ; counters values give $\sum_T (\alpha)$ and E is the sum of all energies for all modules in the distributed processing system 10 dissipated within T. Average system power dissipation W = E/T. In the preferred embodiment, measured and probabilistic energy consumption is calculated and the average power dissipation is derived from the energy consumption over period T. In most cases, energy consumption information will be more readily available. However, it would also be possible to calculate the power dissipation from measured and probabilistic power consumption.

Figure 4b is a flow chart describing operation of a second embodiment of the power management tasks 38. The flow of Figure 4b is the same as that of Figure 41, except when the scenario construction algorithm is invoked (new task, task delete, real time event) in step 50, instead of choosing one new scenario, *n* different scenarios that match the performances constraints can be pre-computed in advance and stored in steps 54 and 59, in order to reduce the number of operations within the dynamic loop and provide faster adaptation if the power computed in the tracking loop leads to current scenario rejection in block 74. In Figure 4b, if the

scenario is rejected, another pre-computed scenario is selected in block 65. Otherwise the operation is the same as shown in Figure 4a.

Figures 5 – 8 illustrate the operation of various blocks of Figure 3 in greater detail. The build system block 54 is shown in Figure 5. In this block, a task list 52, a task model 56, and a list of possible task degradations 58 are used to generate a scenario. The task list is dependent upon which tasks are to be executed on the distributed processing system 10. In the example of Figure 5, three tasks are shown: MPEG4 decode, wireless modem data receive and keyboard event monitor. In an actual implementation, the tasks could come from any number of sources. The task model sets forth conditions which must be taken in consideration in defining the scenario, such as latency and priority constraints, data flow, initial energy estimates, and the impact of degradations. Other conditions could also be used in this block. The output of the build system scenario block is a scenario 80, which associates the various tasks with the modules and assigns priorities to each of the tasks. In the example shown in Figure 5, for example, the MPEG4 decode task has a priority of 16 and the wireless modem task has a priority of 4.

The scenarios built in block 54 could be based on a number of different considerations. For example, the scenarios could be built based on providing the maximum performance within the packages thermal constraints. Alternatively, the scenarios could be based on using the lowest possible energy. The optimum scenario could change during operation of a device; for example, with fully charged batteries a device may operate at a maximum performance level. As the power in the batteries diminished below a preset level, the device could operate at the lowest possible power level to sustain operation.

The scenario 80 from block 54 is used by the activities estimate block 60, shown in Figure 6. This block performs a probabilities computation for various parameters that affect power usage in the distributed processing system 10. The probabilistic activities estimate is generated in conjunction with task activity profiles 62 and hardware architectural models 64. The task activity profiles include information on the data access types (load/store) and occurrences for the different memories, code profiles, such as the branches and loops used in the task, and the cycles per instruction for instructions in the task. The hardware architectural model 64 describes in some way the impact of the task activity profiles 62 on the system latencies, that will permit

computation of estimated hardware activities (such as processor run/wait time share). This model takes into account the characteristics of the hardware on which the task will be implemented, for example, the sizes of the caches, the width of various buses, the number of I/O pins, whether the cache is write-through or write back, the types of memories used (dynamic, static, flash, and so on) and the clock speeds used in the module. Typically, the model can consist of a family of curves that represent MPU and DSP effective frequency variations with different parameters, such as data cacheable/non-cacheable, read/write access shares, number of cycles per instruction, and so on. In the illustrated embodiment of Figure 6, values for the effective frequency of each module, the number of memory accesses, the I/O toggling rates and the DMA flow are calculated. Other factors that affect power could also be calculated.

The power compute block 66 is shown in Figure 8. In this block, the probabilistic activities from block 60 or the measured activities from block 76 are used to compute various energy values and, hence, power values over a period T. The power values are computed in association with hardware power profiles, which are specific to the hardware design of the distributed processing system 10. The hardware profiles could include a Cpd for each module, logic design style (D-type flip-flop, latches, gated clocks and so on), supply voltages and capacitive loads on the outputs. Power computations can be made for integrated modules, and also for external memory or other external devices.

Activity measure and monitor block 76 is shown in Figure 8. Counters are implemented throughout the distributed processing system 10 to measure activities on the various modules, such as cache misses, TLB (translation lookaside buffer) misses, non-cacheable memory accesses, wait time, read/write requests for different resources, memory overhead and temperature. The activity measure and monitor block 76 outputs values for the effective frequency of each module, the number of memory accesses, the I/O toggling rates and the DMA flow. In a particular implementation, other values may also be measured. The output of this block is sent to the power compute block 66.

Figure 9 illustrates and example of a distributed processing system 10 using power/energy management software. In this example, the distributed processing system 10 includes a MPU 12, executing an OS, and two DSPs 14 (individually referenced as DSP1 14a and DSP2 14b), each

executing a respective RTOS. Each module is executing a monitor task 82, which monitors the values in various activity counters 78 throughout the distributed processing system 10. The power compute task is executed on DSP 14a. The various monitor tasks retrieve data from associated activity counters 78 and pass the information to DSP 14a to calculate a power value based on measured activities. The power management tasks, such as power compute task 84 and monitor task 82, can be executed along with other application tasks.

In the preferred embodiment, the power management tasks 38 and profiles 36 are implemented as JAVA class packages in a JAVA real-time environment.

The present invention provides significant advantages over the prior art. First, it provides for a fully dynamic power management. As the tasks executed in the distributed processing system 10 change, the power management can build new scenarios to ensure that thresholds are not exceeded. Further, as environmental conditions change, such as battery voltages dropping, the power management software can re-evaluate conditions and change scenarios, if necessary. For example, if the battery voltage (supply voltage) dropped to a point where Vdd could not be sustained at its nominal value, a lower frequency could be established, which would allow operation of the distributed processing system 10 at a lower Vdd. New scenarios could be built which would take the lower frequency into account. In some instances, more degradations would be introduced to compensate for the lower frequency. However, the lower frequency could provide for continued operation of the device, despite supply voltages that would normally be insufficient. Further, in situations where a lower frequency was acceptable, the device could operate at a lower Vdd (with the availability of a switched mode supply) in order to conserve power during periods of relatively low activity.

The power management software is transparent to the various tasks that it controls. Thus, even if a particular task does not provide for any power management, the power management software assumes responsibility for executing the task in a manner that is consistent with the power capabilities of the distributed processing system 10.

The overall operation of the power management software can be used with different hardware platforms, with different hardware and tasks accommodated by changing the profiles 36.

Although the Detailed Description of the invention has been directed to certain exemplary embodiments, various modifications of these embodiments, as well as alternative embodiments, will be suggested to those skilled in the art. The invention encompasses any modifications or alternative embodiments that fall within the scope of the Claims.

CLAIMS

1. A method for controlling the execution of tasks in a processor comprising a plurality of processing modules, comprising the steps of:

calculating consumption information based on probabilistic values for activities associated with the tasks;

executing the tasks on said plurality of processing modules responsive to said consumption information.

2. The method of claim 1 and further comprising the steps of:

monitoring actual activity occurrences in processing modules; and

modifying the execution of the tasks based on said monitoring step.

3. The method of claim 1 or 2 wherein said executing step comprises the step of executing the tasks on said plurality of processing modules responsive to said consumption information in order to provide the maximum performance within thermal constraints associated with the processing system.

4. The method of claim 1 or 2 wherein said executing step comprises the step of executing the tasks on said plurality of processing modules responsive to said consumption information in order to execute the tasks using the lowest possible energy consumption.

5. The method of claim 1 or 2 wherein said calculating step comprises the steps of:

generating a task allocation scenario;

estimating the activities for task allocation scenario;

computing the consumption associated with said activities.

6. The method of claim 5 wherein said step of generating a task allocation scenario comprises the step of receiving a task list describing the tasks to be executed and a task model describing the tasks.

7. The method of claim 6 wherein the task model includes initial estimates for each task.

8. The method of claim 7 wherein the task model further includes priority constraints associated with the tasks.

9. The method of claim 8 wherein said task model includes information regarding possible degradations associated with one or more of the tasks in said task list.

10. The method of claim 5 wherein said computing step comprises the step of computing the energy consumption associated with said activities.

11. The method of claim 5 wherein said computing step comprises the step of computing the power consumption associated with said activities.

12. A processing device comprising:

one or more processing modules for executing a plurality of tasks, said processing subsystems executing a power management function for:

calculating consumption information based on probabilistic values for activities associated with the tasks;

controlling the execution of the tasks on said processing modules responsive to said consumption information.

13. The processing device of claim 12 and further comprising counters for measuring activity occurrences and wherein said power management function further:

monitors said counters; and

modifies the execution of the tasks based on values in said counters.

14. The processing device of claim 12 or 13 wherein said power management function controls the execution of tasks on the processing modules responsive to said consumption information in order to provide the maximum performance within thermal constraints associated with the processing system.

15. The processing device of claim 12 or 13 wherein said power management function controls the execution of tasks on said processing modules responsive to said consumption information in order to execute the tasks using the lowest possible energy consumption.

16. The processing device of claim 12 or 13 wherein said power management function calculates the consumption information by:

generating a task allocation scenario;

estimating the activities for said task allocation scenario;

computing the consumption associated with said activities.

17. The processing device of claim 16 wherein said power management function generates a task allocation scenario by receiving a task list describing the tasks to be executed and a task model describing the tasks.

18. The processing device of claim 17 wherein the task model includes initial estimates for each task.

19. The processing device of claim 18 wherein the task model further includes priority constraints associated with the tasks.

20. The processing device of claim 19 wherein said task model includes information regarding possible degradations associated with one or more of the tasks in said task list.

21. The processing device of claim 16 wherein said power management function computes the consumption by computing the energy consumption associated with said activities.

22. The processing device of claim 16 wherein said power management function computes the consumption by computing the power consumption associated with said activities.

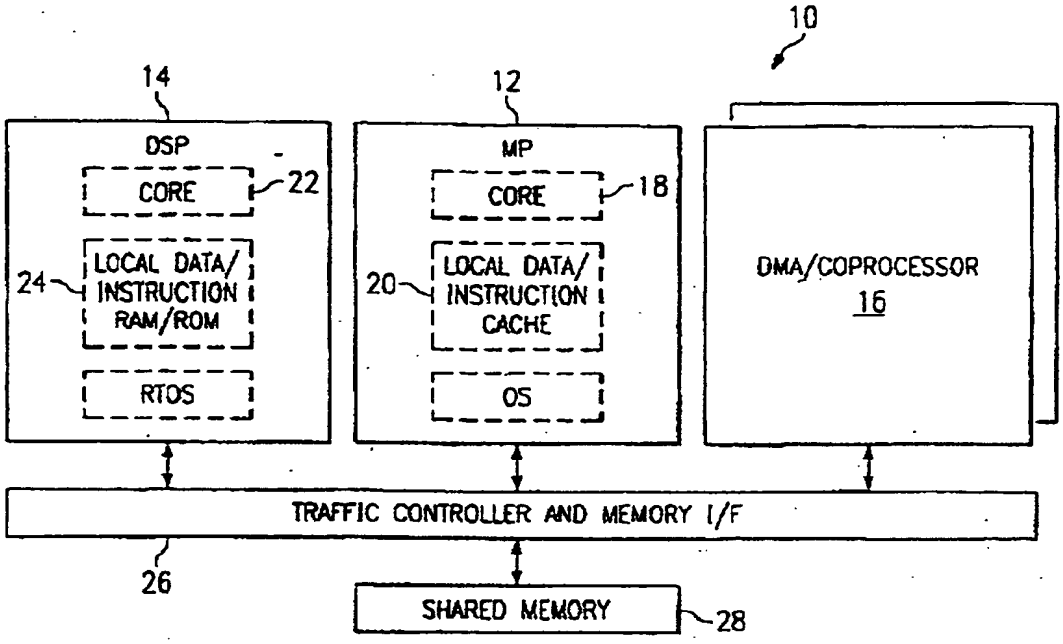

10
14
12
DSP
CORE
22
MP
CORE
18
DMA/COPROCESSOR
16
24
LOCAL DATA/
INSTRUCTION
RAM/ROM
20
LOCAL DATA/
INSTRUCTION
CACHE
RTOS
OS
TRAFFIC CONTROLLER AND MEMORY I/F
26
SHARED MEMORY
28

FIG. 1

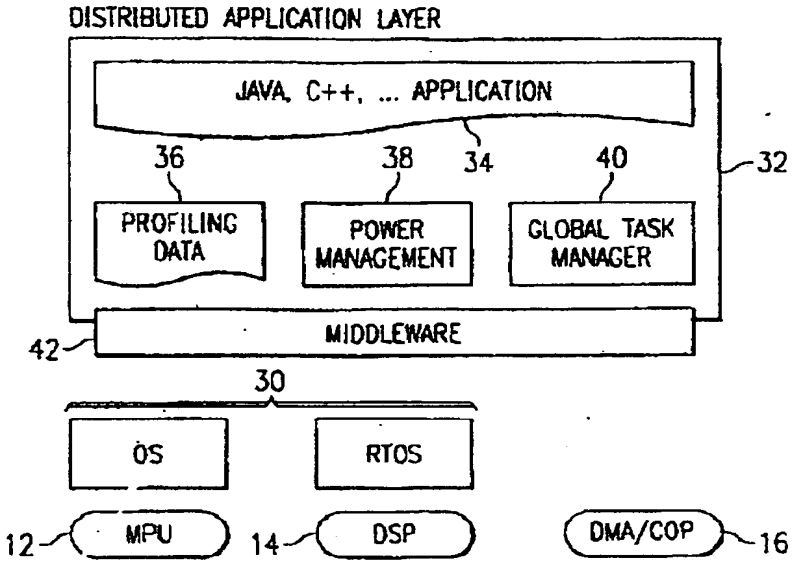

DISTRIBUTED APPLICATION LAYER
JAVA, C++, ... APPLICATION
34
36
38
40
32
PROFILING
DATA
POWER
MANAGEMENT
GLOBAL TASK
MANAGER
MIDDLEWARE
42
30
OS
RTOS
12
MPU
14
DSP
DMA/COP
16

FIG. 2

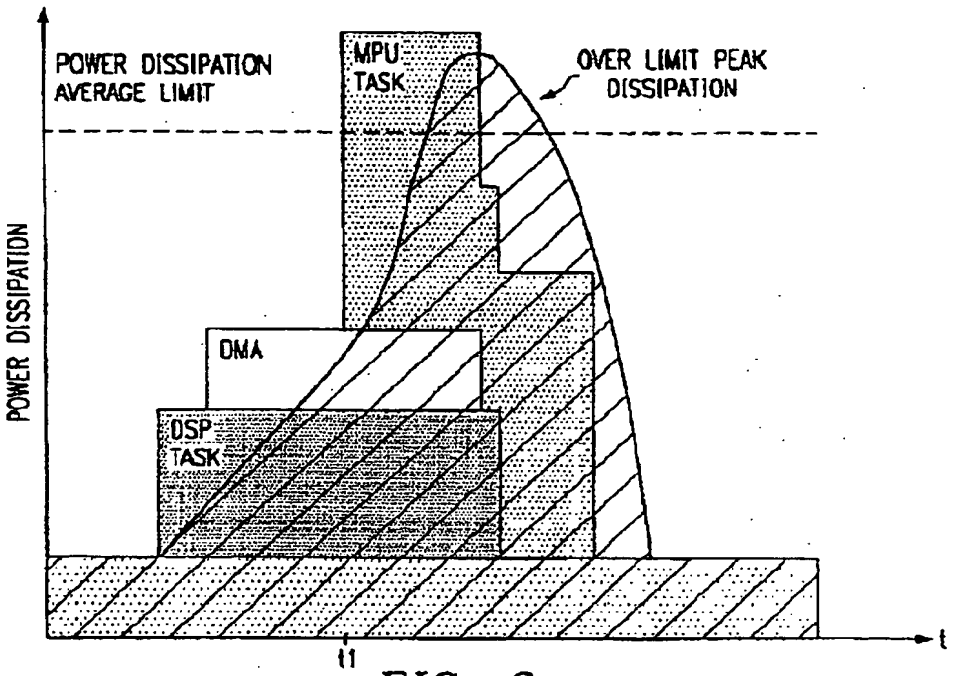

POWER DISSIPATION
AVERAGE LIMIT
MPU TASK
OVER LIMIT PEAK DISSIPATION
DMA
DSP TASK
POWER DISSIPATION
t
t1

FIG. 3a

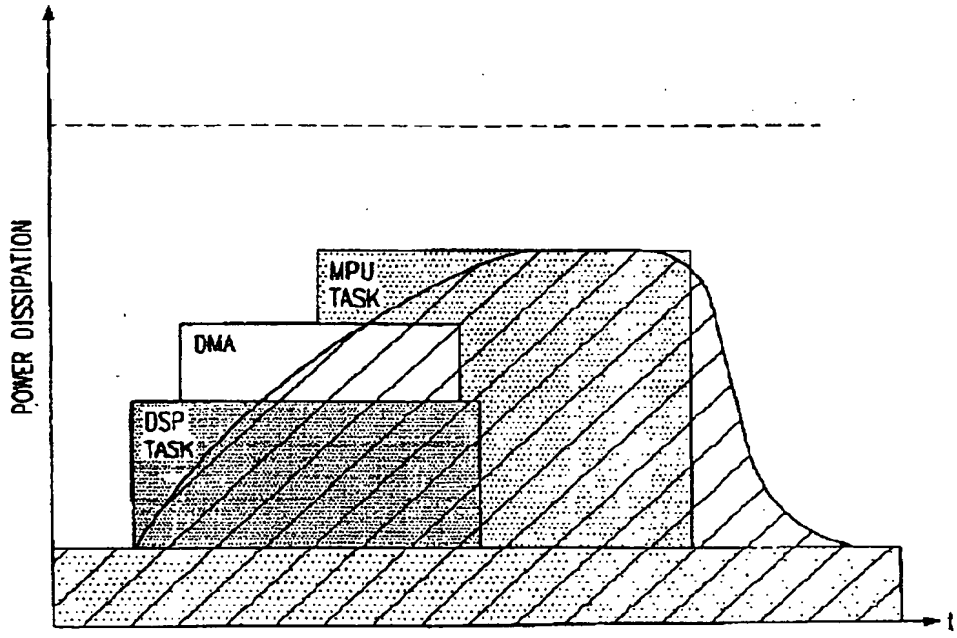

MPU TASK
DMA
DSP TASK
POWER DISSIPATION
t

FIG. 3b

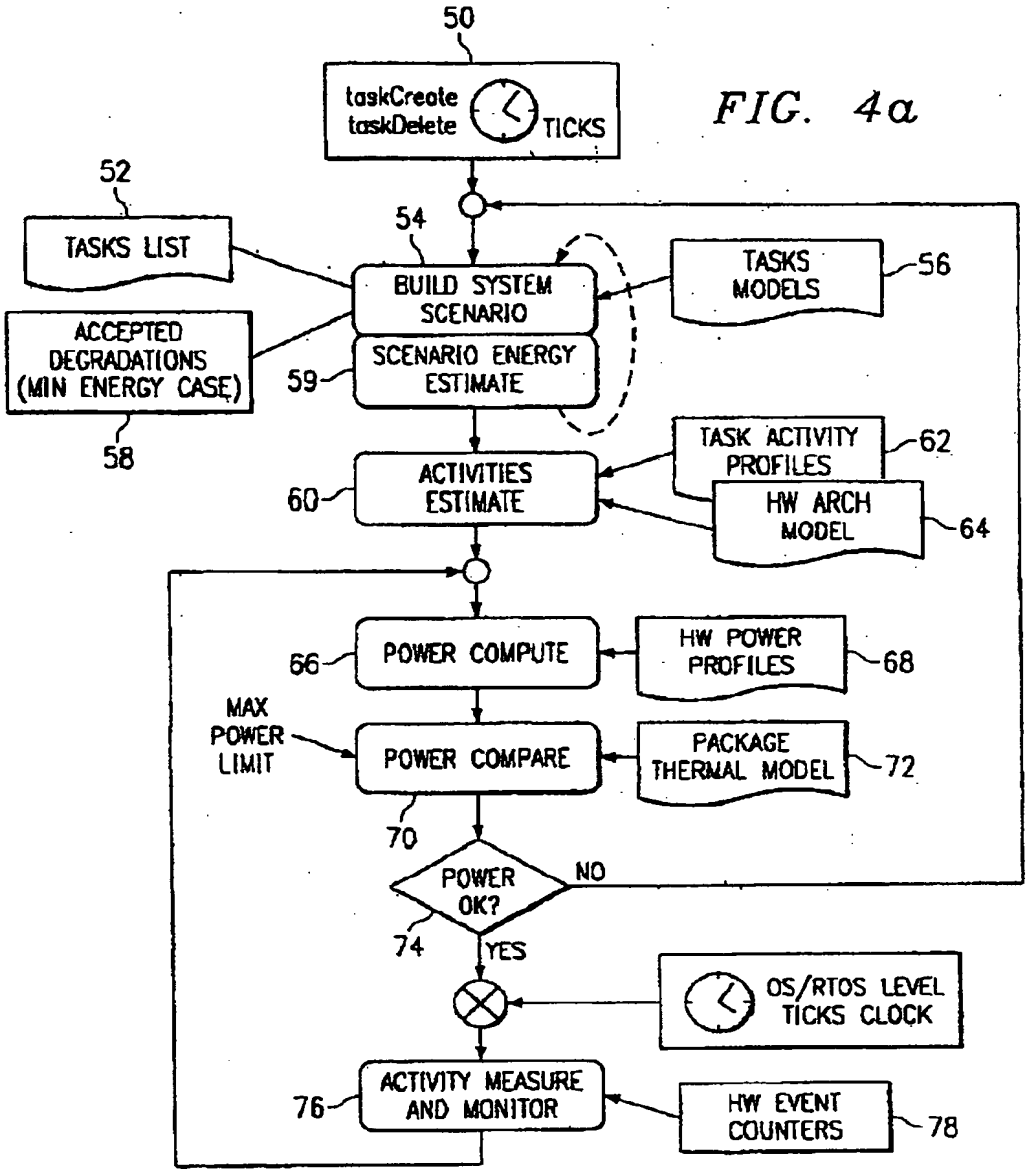

50
taskCreate
taskDelete
TICKS
52
TASKS LIST
54
BUILD SYSTEM SCENARIO
TASKS MODELS
56
ACCEPTED DEGRADATIONS (MIN ENERGY CASE)
58
59
SCENARIO ENERGY ESTIMATE
TASK ACTIVITY PROFILES
62
60
ACTIVITIES ESTIMATE
HW ARCH MODEL
64
66
POWER COMPUTE
HW POWER PROFILES
68
MAX POWER LIMIT
POWER COMPARE
PACKAGE THERMAL MODEL
72
70
POWER OK?
NO
74
YES
OS/RTOS LEVEL TICKS CLOCK
76
ACTIVITY MEASURE AND MONITOR
HW EVENT COUNTERS
78

FIG. 4a

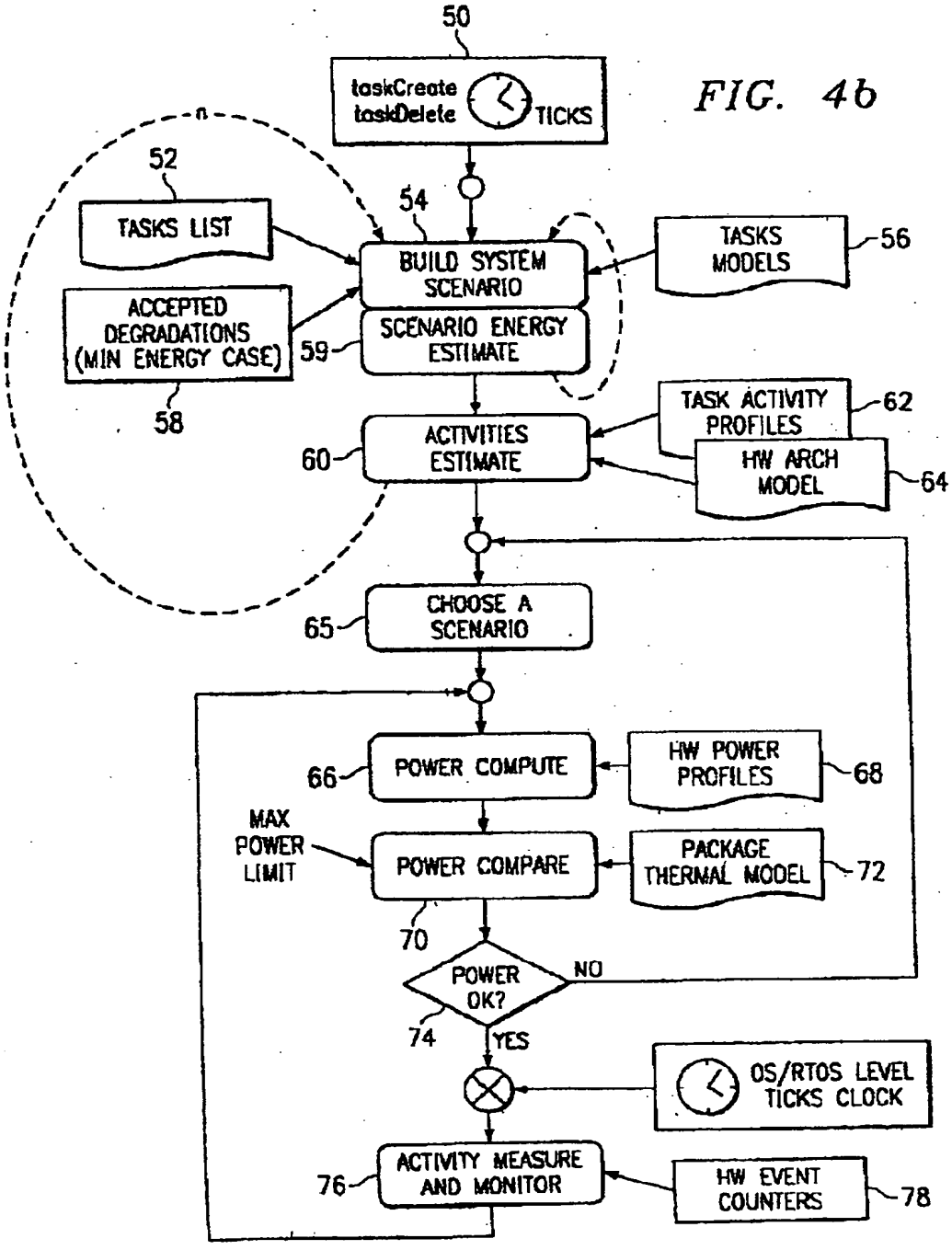

50
taskCreate
taskDelete
TICKS
52
TASKS LIST
54
BUILD SYSTEM SCENARIO
TASKS MODELS
56
ACCEPTED DEGRADATIONS (MIN ENERGY CASE)
59
SCENARIO ENERGY ESTIMATE
58
TASK ACTIVITY PROFILES
62
60
ACTIVITIES ESTIMATE
HW ARCH MODEL
64
65
CHOOSE A SCENARIO
66
POWER COMPUTE
HW POWER PROFILES
68
MAX POWER LIMIT
POWER COMPARE
PACKAGE THERMAL MODEL
72
70
POWER OK?
NO
74
YES
OS/RTOS LEVEL TICKS CLOCK
76
ACTIVITY MEASURE AND MONITOR
HW EVENT COUNTERS
78

FIG. 4b

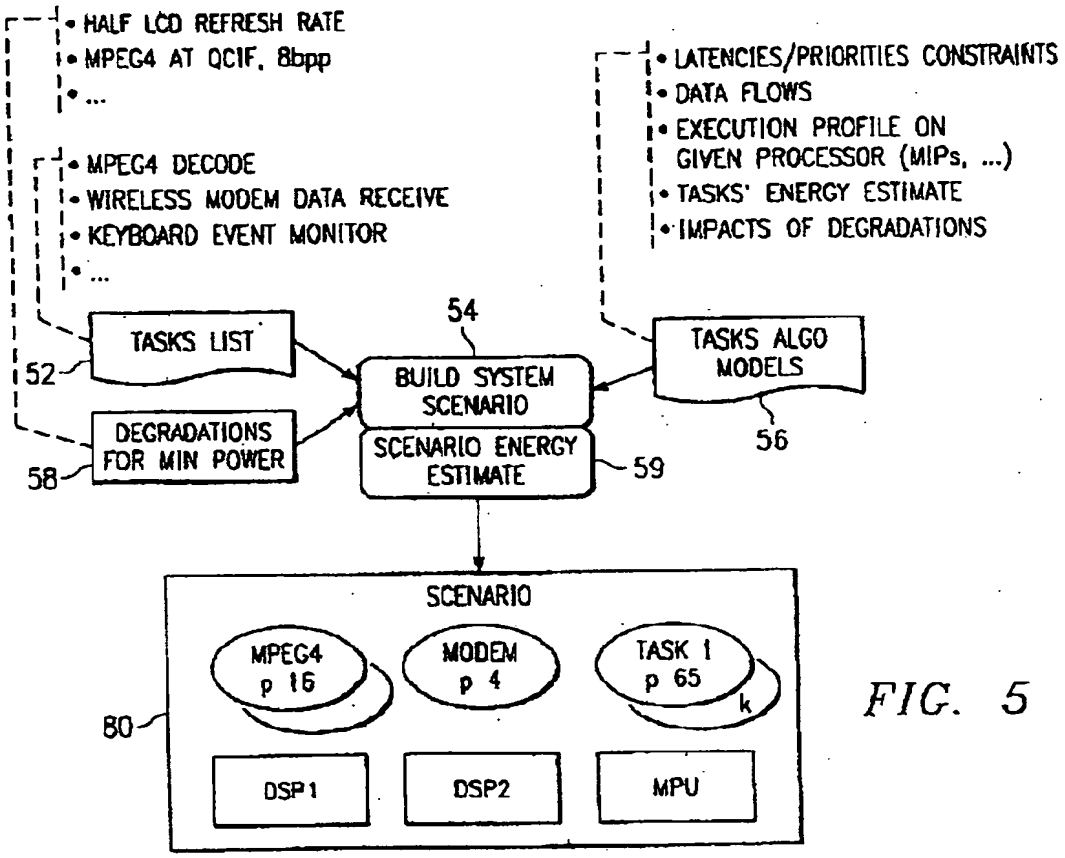

• HALF LCD REFRESH RATE
• MPEG4 AT QCIF, 8bpp
• ...
• MPEG4 DECODE
• WIRELESS MODEM DATA RECEIVE
• KEYBOARD EVENT MONITOR
• ...
• LATENCIES/PRIORITIES CONSTRAINTS
• DATA FLOWS
• EXECUTION PROFILE ON GIVEN PROCESSOR (MIPs, ...)
• TASKS' ENERGY ESTIMATE
• IMPACTS OF DEGRADATIONS
TASKS LIST
52
54
BUILD SYSTEM SCENARIO
TASKS ALGO MODELS
56
DEGRADATIONS FOR MIN POWER
58
SCENARIO ENERGY ESTIMATE
59
SCENARIO
MPEG4 p 16
MODEM p 4
TASK 1 p 65
k
80
DSP1
DSP2
MPU

FIG. 5

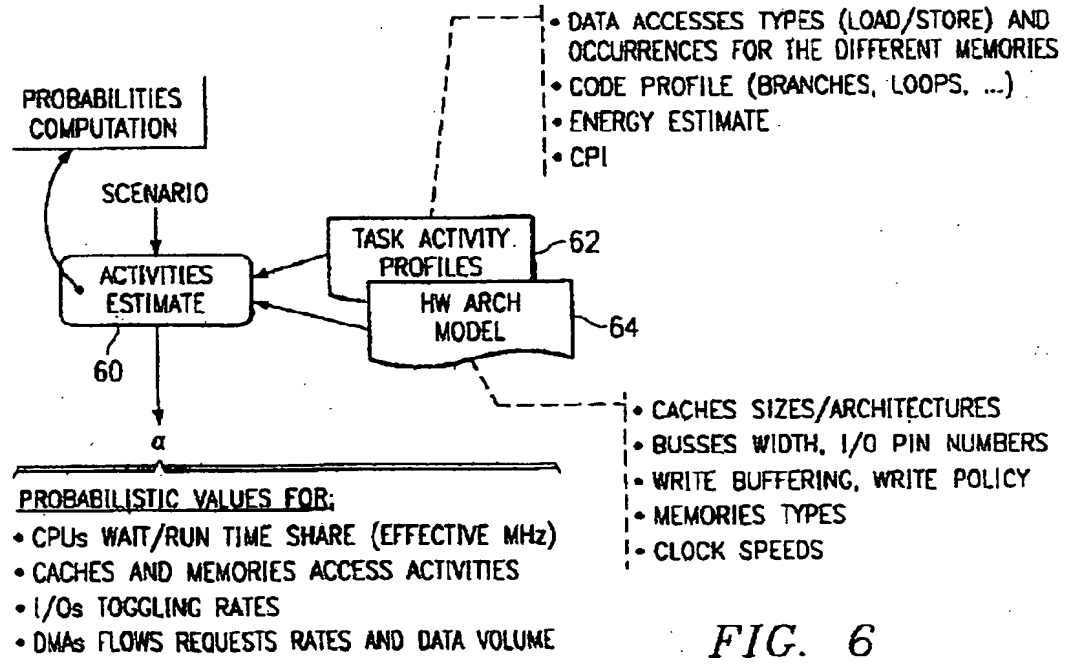

• DATA ACCESSES TYPES (LOAD/STORE) AND OCCURRENCES FOR THE DIFFERENT MEMORIES
• CODE PROFILE (BRANCHES, LOOPS, ...)
• ENERGY ESTIMATE
• CPI
PROBABILITIES COMPUTATION
SCENARIO
TASK ACTIVITY PROFILES
62
ACTIVITIES ESTIMATE
HW ARCH MODEL
64
60
a
PROBABILISTIC VALUES FOR:
• CPUs WAIT/RUN TIME SHARE (EFFECTIVE MHz)
• CACHES AND MEMORIES ACCESS ACTIVITIES
• I/Os TOGGLING RATES
• DMAs FLOWS REQUESTS RATES AND DATA VOLUME
• CACHES SIZES/ARCHITECTURES
• BUSSES WIDTH, I/O PIN NUMBERS
• WRITE BUFFERING, WRITE POLICY
• MEMORIES TYPES
• CLOCK SPEEDS

FIG. 6

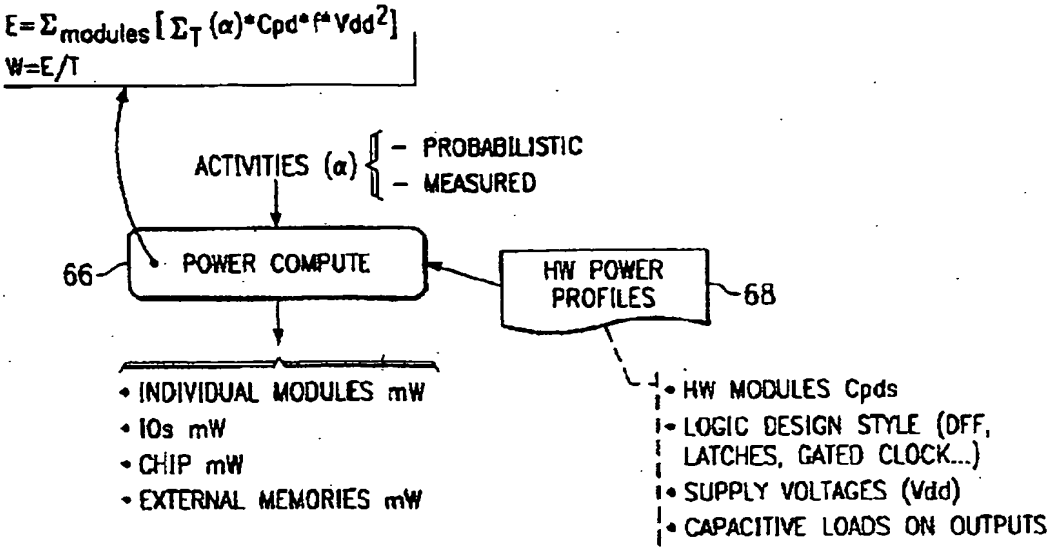

E=Σmodules [ΣT (α)*Cpd*f*Vdd2]
W=E/T
ACTIVITIES (α)
- PROBABILISTIC
- MEASURED
66
POWER COMPUTE
HW POWER PROFILES
68
• INDIVIDUAL MODULES mW
• IOs mW
• CHIP mW
• EXTERNAL MEMORIES mW
• HW MODULES Cpds
• LOGIC DESIGN STYLE (DFF, LATCHES, GATED CLOCK...)
• SUPPLY VOLTAGES (Vdd)
• CAPACITIVE LOADS ON OUTPUTS

FIG. 7

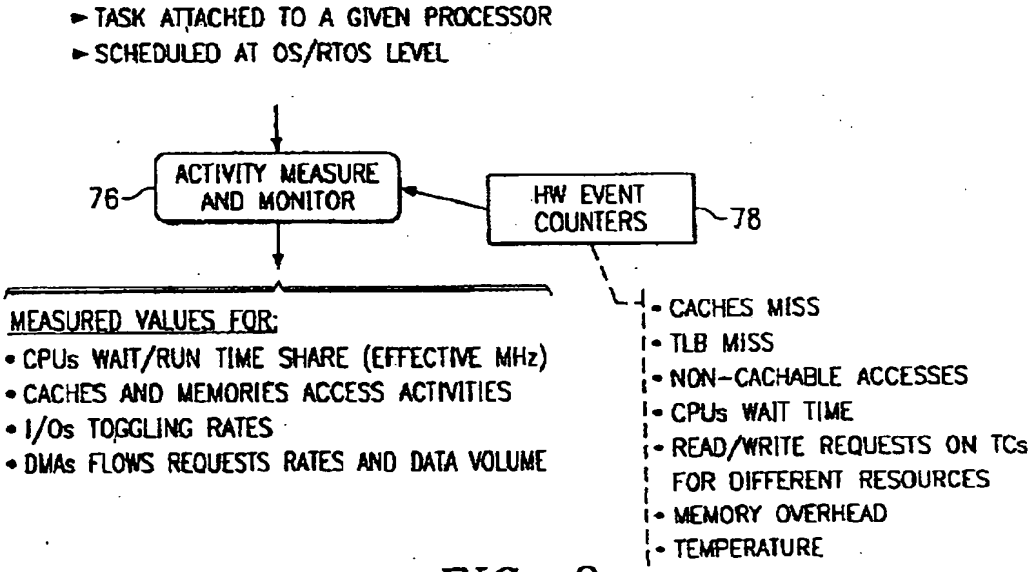

► TASK ATTACHED TO A GIVEN PROCESSOR
► SCHEDULED AT OS/RTOS LEVEL
76
ACTIVITY MEASURE AND MONITOR
HW EVENT COUNTERS
78
MEASURED VALUES FOR:
• CPUs WAIT/RUN TIME SHARE (EFFECTIVE MHz)
• CACHES AND MEMORIES ACCESS ACTIVITIES
• I/Os TOGGLING RATES
• DMAs FLOWS REQUESTS RATES AND DATA VOLUME
• CACHES MISS
• TLB MISS
• NON-CACHABLE ACCESSES
• CPUs WAIT TIME
• READ/WRITE REQUESTS ON TCs FOR DIFFERENT RESOURCES
• MEMORY OVERHEAD
• TEMPERATURE

FIG. 8

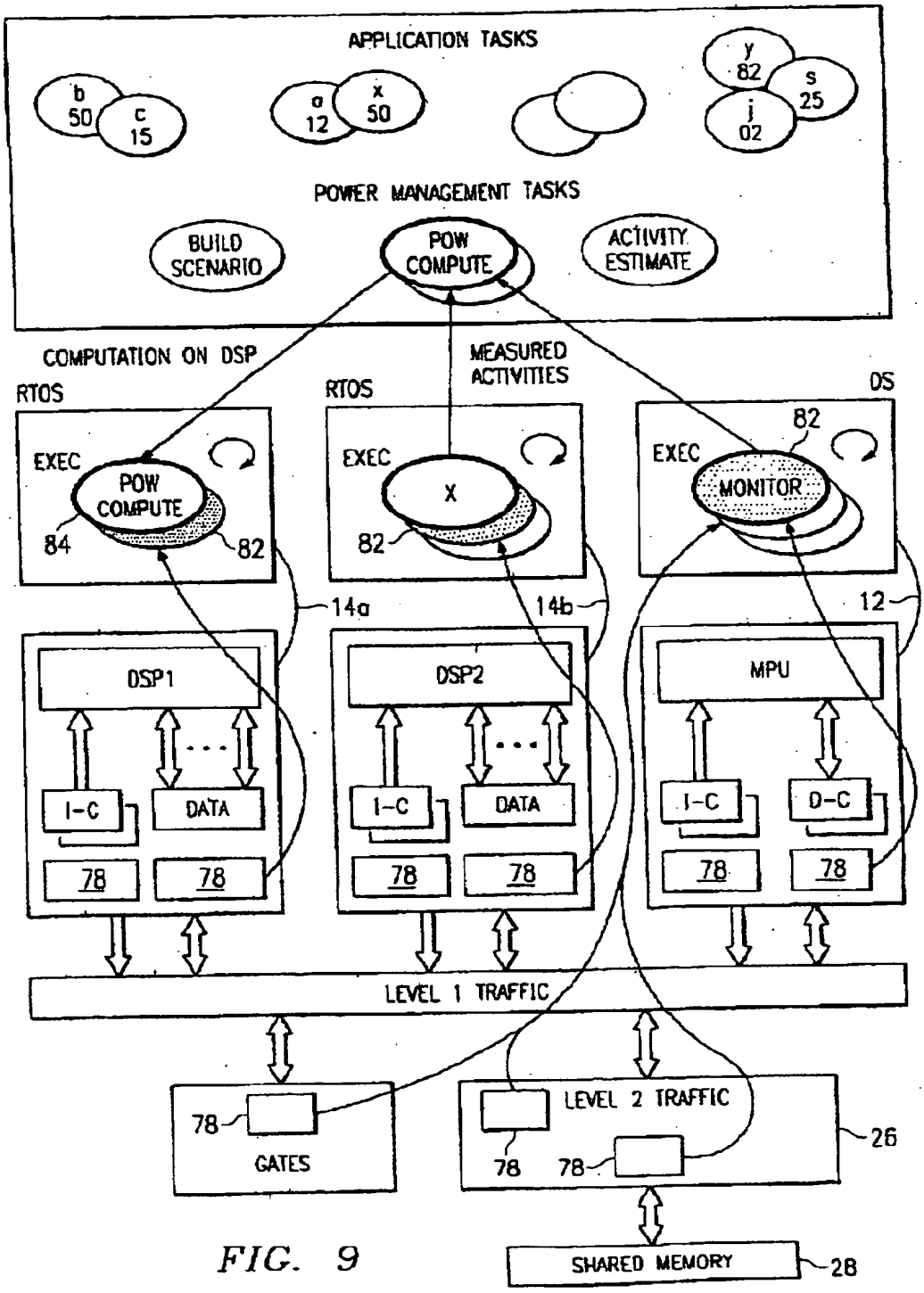

APPLICATION TASKS
b 50
c 15
a 12
x 50
y 82
s 25
j 02
POWER MANAGEMENT TASKS
BUILD SCENARIO
POW COMPUTE
ACTIVITY ESTIMATE
COMPUTATION ON DSP
MEASURED ACTIVITIES
RTOS
RTOS
OS
EXEC
POW COMPUTE
84
82
EXEC
X
82
EXEC
82
MONITOR
14a
14b
12
DSP1
DSP2
MPU
I-C
DATA
I-C
DATA
I-C
D-C
78
78
78
78
78
78
LEVEL 1 TRAFFIC
78
GATES
LEVEL 2 TRAFFIC
78
78
26
SHARED MEMORY
28

FIG. 9

1 Abstract

ABSTRACT OF THE DISCLOSURE

A distributed processing system (10) includes a plurality of processing modules, such as MPUs (12), DSPs (14), and coprocessors/DMA channels (16). Power management software (38) in conjunction with profiles (36) for the various processing modules and the tasks to executed are used to build scenarios which meet predetermined power objectives, such as providing maximum operation within package thermal constraints or using minimum energy. Actual activities associated with the tasks are monitored during operation to ensure compatibility with the objectives. The allocation of tasks may be changed dynamically to accommodate changes in environmental conditions and changes in the task list.